Eötvös Loránd University

Faculty of Informatics

Dept. of Software Technology And Methodology

Predictive analysis in Healthcare

Supervisor:
Molnár Bálint
Dr. Habil. Assistant Lecturer

Author:
Tshifaro Justin Ndivhuwo
Computer Science MSc

Budapest, 2023

Computer Science MSc
1st Semester, 2022-2023 Academic Year

Thesis Topic Description

Submission deadline: 2022.09.30.

Student's Data:

Name: Tshifaro Ndivhuwo Justin
Neptun Code: SUQG0J

Course Data:

Student's Major: Computer Science MSc
Semester & Academic Year of Student's Admission: 2021/22/1

A brief description of the thesis topic:

(Consulting with your potential supervisor, give a synopsis of your planned thesis in a few sentences.)

Predictive analysis in Health care
The goal of predictive analytics is to determine the likelihood of future outcomes based on historical data. It does this by using data, statistical algorithms, and machine-learning approaches. The objective is to provide the best prediction of what will occur in the future, rather than simply knowing what has already occurred.
Predicting the future with the aid of machine learning algorithms is no longer a challenging task, particularly in the field of medicine where it is now possible to predict diseases and anticipate a cure. An overview of the development of big data in the healthcare system will be provided in this paper, and a learning algorithm will be applied to a set of health data. This dissertation aims to predict chronic diseases and mental health using Clustering and Classification models. Both Disadvantages and Advantages Disadvantages will be discussed.

During the experimental design and development, adequate methodologies are used (Design Science Research, Software Case Study, Data Science data and system modeling, etc.). The developed model will be verified and validated by domain-specific metrics.

Thesis Supervisor: Dr. Bálint Molnár

Specialisation:
Data Science

Subject related to the topic of the thesis:

Intorduction to Data Science

Advanced Machine Learning

Open-source Technologies for Real-time Data Analytics

Contents

Abstract

Chapter 1

Introduction

Chapter 2

Background and Problem Understanding

Chapter 3

Methodology

This chapter aims to describe and fully understand the research methods used. This research's primary objectives are to build a predictive machine learning model, train them on health-related datasets, and optimize its hyper-parameters utilizing swarm intelligence techniques, such as Particle Swarm Optimization (PSO), and Firefly Algorithm. The models and algorithms deployed for this project are thoroughly described below.

3.1 Classification models

3.1.1 XGBoost algorithm

Extreme Gradient Boosting, often known as XGBoost, is an efficient machine learning technique that is frequently applied in data science and machine learning applications. The Gradient Boosting Decision Tree (GBDT) method, on which it is based, is known for its effectiveness, scalability, and accuracy [1].

The XGBoost algorithm operates by assembling a group of decision trees. It creates decision trees iteratively and combines them to produce a final prediction. XGBoost detects sections of the data where the model performs badly in each iteration and concentrates on improving those areas in the next iteration. This process is repeated until the model reaches the appropriate level of precision.

XGBoost is popular among data scientists and machine learning practitioners due to a number of essential features. Its ability to handle missing data is one of its most crucial qualities. It can also handle both numerical and categorical data

and can discover the optimum way to represent categorical variables in the model automatically.

XGBoost is also well-known for its scalability and efficiency. It is designed to take advantage of parallel processing and can handle massive datasets efficiently. XGBoost has also been tuned for performance, including a variety of strategies that reduce memory usage and calculation time.

The adoption of regularization techniques is one of the reasons why XGBoost is so accurate. These strategies prevent overfitting and improve model generalization. XGBoost additionally includes a set of hyper-parameters that can be tuned to improve the model's performance.

To understand the XGBoost well, it is crucial to understand the decision tree first. Figure 3.1 shows the structure of a Decision Tree and figure 3.1 Shows the structure of XGBoost algorithm

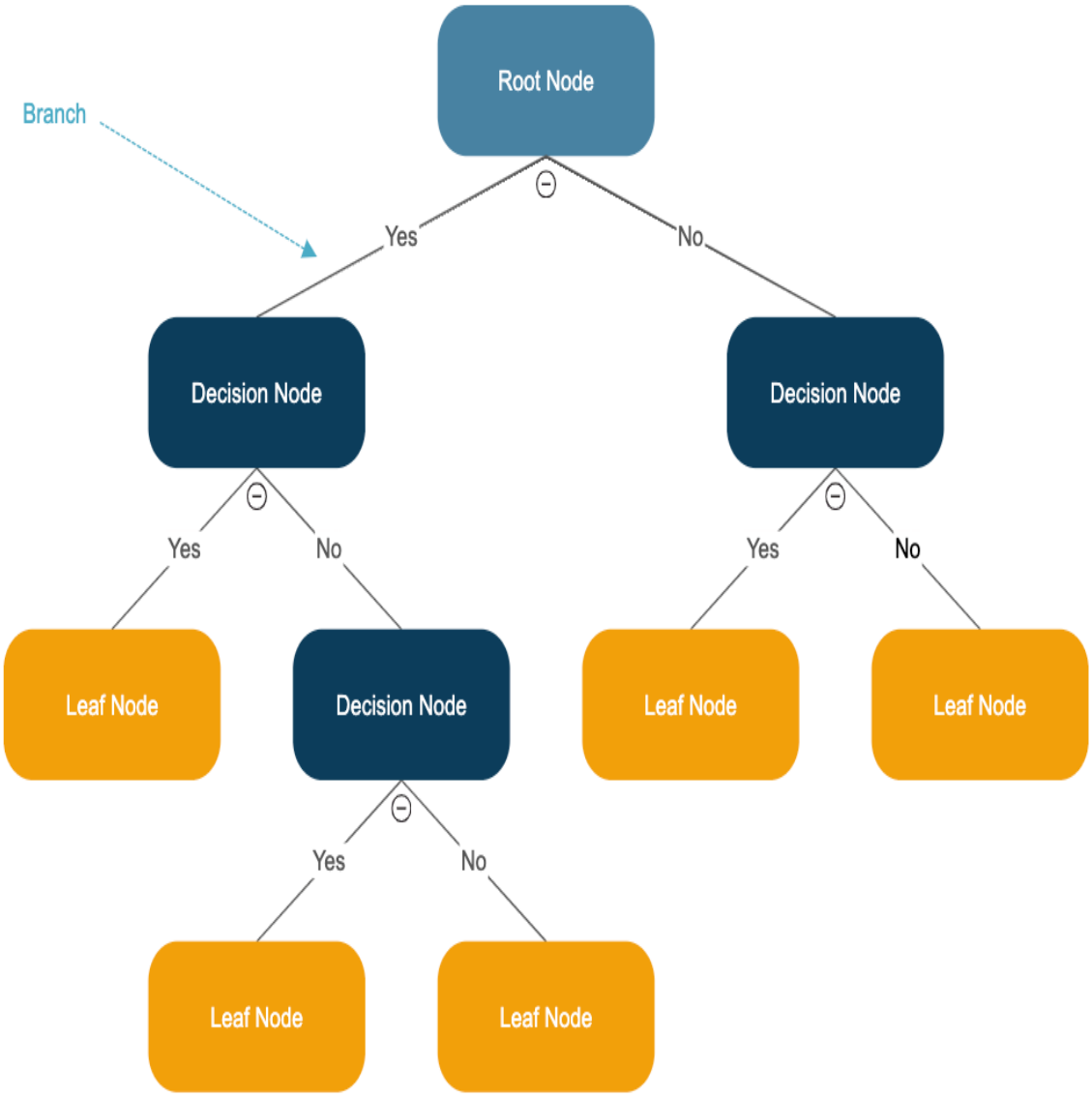

Figure 3.1: Decision Tree Structure
[2]

XGBoost Hyper-parameters

A hyperparameter is a machine learning parameter whose value is determined before training a learning algorithm.

1. learning_rate This influences the size of the step for each iteration of the boosting procedure. Lower values will make the model more conservative but converge may take longer iterations. Smaller learning rates generally require more trees to be added to the model. It is often referred to as the "shrinkage factor," and it is typically set to a low number, such as 0.1 or 0.01, to guarantee that the model converges slowly and smoothly.

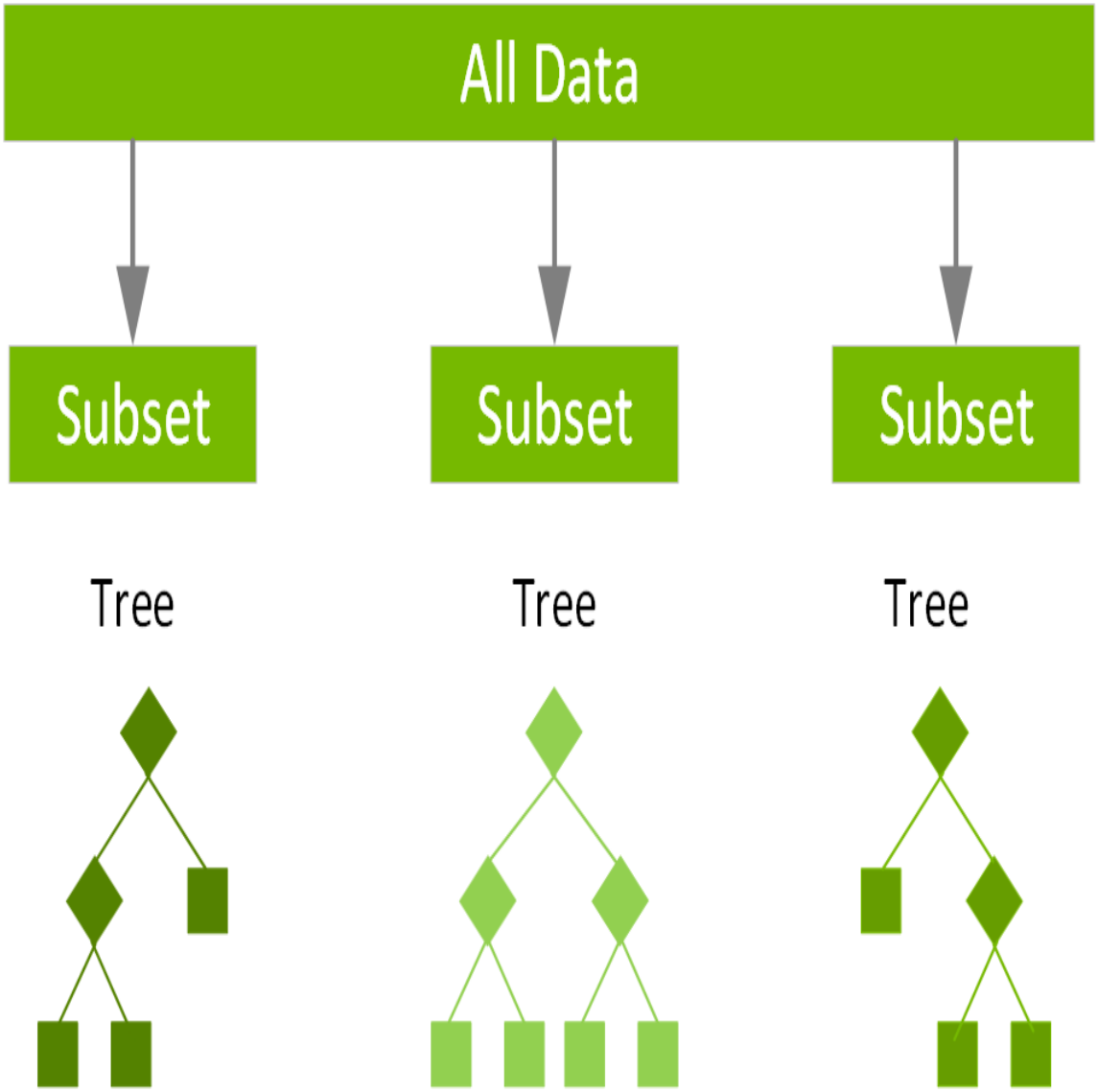

Figure 3.2: Decision Tree Structure
[3]

2. max_depth This regulates each tree in the ensemble's maximum depth. However, deeper trees run the risk of overfitting the data and capturing more complicated interactions. In simple words, it controls the size of decision trees.

3. Subsample This regulates how much of the training data is utilized to create each tree. Smaller numbers will increase the model's resistance to noise, but they can compromise its capacity to detect complex trends in the data. Greater and more complicated models are produced by greater subsample values, which might result in overfitting. This value is typically set between 0.5 and 1.

4. colsample_bytree The colsample_bytree parameter determines how many fea-

tures are used for each tree. A lower colsample_bytree number produces smaller and less complex models, which can aid in the prevention of overfitting. A higher colsample_bytree value leads to larger and more complicated models, which can result in overfitting.

5. n_estimators This determines how many trees are in the ensemble. More trees can improve model accuracy while also increasing the danger of overfitting.

6. min_child_weight This specifies the minimum instance weight required to split a node. Higher values will reduce overfitting and make the model more conservative.

7. gamma This regulates the minimal loss function decrease necessary to split a node. Higher values will reduce overfitting and make the model more conservative.

These are just a few of the many parameters that may be tweaked in XGBoost. There are numerous hyper-parameters, but selecting the optimal mix of parameters for a specific problem takes rigorous experimentation and adjustment, and for these experiments, we chose the hyper-parameters listed above. Table 3.1 shows the recommended and default value of XGBoost Hyper-parameters.It should be noted that these recommended values are not always ideal for every dataset and problem and that a hyperparameter tuning process is recommended to determine the best values for your unique instance.

Hyper-parameter	Default Value	Recommended Range
$n_estimators$	100	$100 - 1000$
max_depth	6	$3 - 10$
$learning_rate$	0.3	$0.01 - 0.2$
min_child_weight	1	$1 - 10$
subsample	1	$0.5 - 1$
$colsample_bytree$	1	$0.5 - 1$
gamma	0	$0 - 5$

Table 3.1: XGBoost default Hyper-parameters

Chapter 4

Experiments and Results

Chapter 5

Conclusion

Bibliography

[1] T. Chen and Guestrin. "XGBoost: A Scalable Tree Boosting System." In: *XGBoost* (). URL: https://doi.org/10.1145/2939672.2939785.

[2] Smartdraw. "Decision Tree". In: *J. Object Oriented Program.* 1.1 (Jan. 2023). URL: https://www.smartdraw.com/decision-tree/.

[3] Jason Brownlee. "A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning". In: *A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning* (). URL: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/.